

PATENT ABSTRACTS OF JAPAN

(11)Publication number : 10-171662

(43)Date of publication of application : 26.06.1998

(51)Int.Cl.

G06F 9/445

G06F 13/00

(21)Application number : 08-340529

(71)Applicant : HITACHI LTD

(22)Date of filing : 05.12.1996

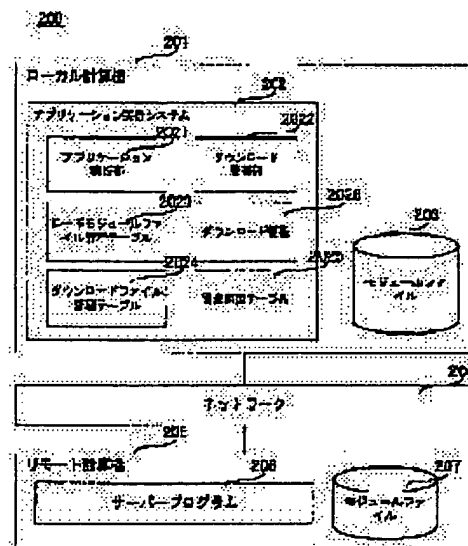
(72)Inventor : MIZOTE YUJI
TOMINAGA MASASUKE
MASUISHI TETSUYA
HONDA YURI
SHIMAZAKI KOICHI

(54) APPLICATION EXECUTING METHOD

(57)Abstract

PROBLEM TO BE SOLVED: To reduce the waiting time of a user in a file download mode by downloading files onto the file system of a local computer from the file system of a remote computer in the order of priority of files and controlling them.

SOLUTION: This application executing system 202 includes an application execution part 2021, a download management part 2022, a load module file management table 2023, a download file management table 2024, the download information 2026 and a priority table 2025. The table 2025 is produced to describe the downloading priority of files forming an application based on the downloaded priority information. Then the files are downloaded onto the file system of a local computer 201 from the file system of a remote computer 205 in the order of priority of files and by making reference to the table 2025 and also to each step. These downloaded files are managed.



LEGAL STATUS

[Date of request for examination]

[Date of sending the examiner's decision of rejection]

[Kind of final disposal of application other than the examiner's decision of rejection or application converted registration]

[Date of final disposal for application]

[Patent number]

[Date of registration]

[Number of appeal against examiner's decision of rejection]

[Date of requesting appeal against examiner's decision of rejection]

[Date of extinction of right]

Copyright (C); 1998,2003 Japan Patent Office

(51)Int.Cl.⁶

識別記号

F I

G 0 6 F 9/445
13/00

3 5 1

G 0 6 F 9/06
13/004 2 0 J
3 5 1 H

審査請求 未請求 請求項の数3 F D (全 22 頁)

(21)出願番号 特願平8-340529

(22)出願日 平成8年(1996)12月5日

(71)出願人 000005108

株式会社日立製作所

東京都千代田区神田駿河台四丁目6番地

(72)発明者 溝手 裕二

神奈川県川崎市幸区鹿島田890番地 株式
会社日立製作所情報・通信開発本部内

(72)発明者 富永 雅介

神奈川県川崎市幸区鹿島田890番地 株式
会社日立製作所情報・通信開発本部内

(72)発明者 増石 哲也

神奈川県川崎市幸区鹿島田890番地 株式
会社日立製作所情報・通信開発本部内

(74)代理人 弁理士 矢島 保夫

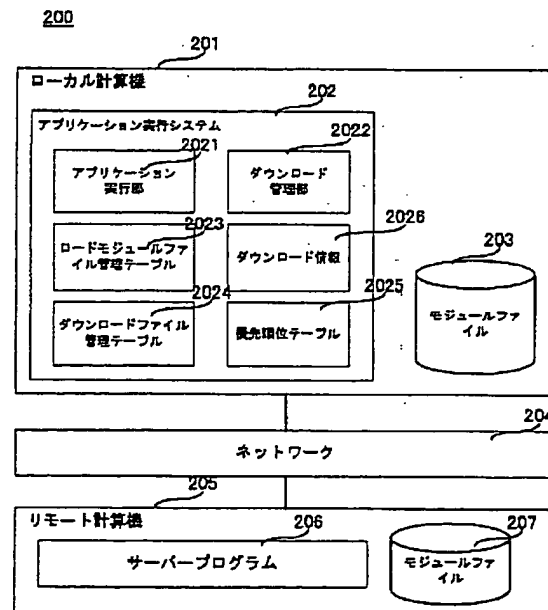
最終頁に続く

(54)【発明の名称】 アプリケーション実行方法

(57)【要約】

【課題】本発明の目的は、アプリケーションを構成するプログラム情報を格納する複数のファイルをリモート計算機からローカル計算機にダウンロードして実行するシステムにおいて、ファイルダウンロードにおけるユーザの待ち時間を削減することである。

【解決手段】特定のファイルのプログラムを実行中に別のファイルが必要になったとき、該ファイルがローカル計算機のファイルシステムに存在しなければ、リモート計算機のファイルシステムからローカル計算機のファイルシステムに該ファイルをダウンロードする処理を行った後、ローカル計算機のファイルシステムにある該ファイルをメモリ中に読み込んで実行する処理と、その処理と並行して行われる処理で、ファイルダウンロードの優先順位を記述する優先順位情報を参照し、その優先順序に基づいて、リモート計算機上のファイルシステムからローカル計算機上のファイルシステムにダウンロードを行うダウンロード処理を行うようにする。



【特許請求の範囲】

【請求項1】複数のファイルから構成されるアプリケーション・プログラムをファイルシステム上に保持するリモート計算機から、前記アプリケーション・プログラムを構成するファイル群をローカル計算機上のファイルシステムにダウンロードし、前記ローカル計算機上で前記ダウンロードしたファイルを内部記憶装置上に格納し、内部記憶装置上に格納したプログラムを解釈して実行するアプリケーション実行方法において、アプリケーションを実行するのに必要なファイルがローカル計算機のファイルシステムに存在するか否かを判定する第1の処理ステップと、

前記第1の処理ステップによりローカル計算機中のファイルシステムに前記ファイルが存在しないことが判明した場合、リモート計算機のファイルシステムからローカル計算機のファイルシステムに前記ファイルをダウンロードする第2の処理ステップと、

前記ファイルをダウンロードしたことを管理する第3の処理ステップと、

前記ファイルをローカル計算機の内部記憶装置に格納するとともに、前記ファイルを内部記憶装置に格納したことを管理する第4の処理ステップと、

前記アプリケーションを構成する複数のファイルのダウンロード処理の優先順位を表す優先順位情報をアプリケーションを構成するファイルの1つ中に含め、前記アプリケーションの実行前に、前記優先順位情報を含むファイルをローカル計算機にダウンロードし、ダウンロードした前記優先順位情報を用いてアプリケーション構成ファイルのダウンロードの優先順位を記述した優先順位テーブルを作成する第5の処理ステップと、

前記第1、第2、第3、および第4の処理ステップと並行して行われる処理であって、前記優先順位テーブルを参照し、その優先順位の順にリモート計算機上のファイルシステムからローカル計算機上のファイルシステムにファイルをダウンロードし、ダウンロードしたファイルを管理する第6の処理ステップとを備えたことを特徴とするアプリケーション実行方法。

【請求項2】前記アプリケーション実行時に、実行対象のファイルを管理する処理ステップと、

前記アプリケーションを構成するファイルの1つ中に、そのアプリケーションを構成する複数ファイル間の依存関係情報を含め、前記依存関係情報および前記実行対象ファイルの管理情報に基づいて、ダウンロードの優先順位をアプリケーションの実行時に動的に決定する処理ステップとをさらに備えたことを特徴とする請求項1に記載のアプリケーション実行方法。

【請求項3】複数のファイルから構成されるアプリケーション・プログラムをファイルシステム上に保持するリモート計算機から、前記アプリケーション・プログラムを構成するファイル群をローカル計算機上のファイルシ

ステムにダウンロードし、前記ローカル計算機上で前記ダウンロードしたファイルを内部記憶装置上に格納し、内部記憶装置上に格納したプログラムを解釈して実行するアプリケーション実行方法において、

前記ローカル計算機上で、前記アプリケーションを構成するファイルのうちの1つを内部記憶装置上に格納して実行する実行ステップと、

前記ローカル計算機上で、前記実行ステップとは別プロセスで並行して、前記アプリケーションを構成する他のファイルのダウンロードを順次行うダウンロードステップとを備えたことを特徴とするアプリケーション実行方法。

【発明の詳細な説明】

【0001】

【発明の属する技術分野】本発明は、アプリケーションを構成するプログラム情報を格納するファイルをリモート計算機のファイルシステムからローカル計算機のファイルシステムにダウンロードして解釈実行するシステム、特に、入出力装置を用いたユーザとの対話的操作を多く含むアプリケーションファイルをダウンロードして実行するシステムにおいて、ファイルダウンロード時のユーザの待ち時間を削減するアプリケーション実行方法に関する。

【0002】

【従来の技術】「NetScapeの機能拡張で存在感薄れるWin95」（日経バイト（1996年4月、日経BP社）、pp166～177）（文献1）に示されているように、近年、世界的規模のネットワーク網であるインターネットとインターネット上の情報開示システムの一つであるWWW（World Wide Web）の発達により、Webサーバ上で開示されたWebページをクライアント側でWebブラウザを用いて見ることはもちろんのこと、Webブラウザ上でマウスを用いた簡単な操作でリモート計算機のファイルをクライアントマシンにダウンロードしたり、ダウンロードするファイルに応じたプログラム（ヘルパープログラムやプラグインと呼ぶ）を起動させることにより、リモート計算機のファイルのデータを読み込んでその場でアプリケーションを実行することも可能になった。

【0003】1つのアプリケーションを構成する複数のファイルをリモート計算機のファイルシステムにおき、それらのファイルをローカル計算機のファイルシステムにダウンロードし、そのファイルをオープンして実行する場合、従来においては、アプリケーション実行前にすべてのファイルをダウンロードするか、実行中にファイルが必要になった時、必要なファイルだけをダウンロードするということが行われていた。

【0004】また、リモート計算機のファイルシステムにあるファイルを、ローカル計算機にネットワーク経由で転送し、そのデータを用いてアプリケーションを実

行するアプリケーション実行環境として、Sun Microsystems, Incが開発したオブジェクト指向型言語Javaを用いて作成したアプリケーションをJava対応のWebブラウザ上で実行する場合がある(上記文献1)。ただし、Javaは、リモート計算機上のアプリケーションファイルをローカル計算機のファイルシステムに格納してからファイルオープンして実行するのではなく、ローカル計算機のメモリに直接読みこんで実行するものである。Javaでは、アプリケーションの実行単位は“アプレット”と呼ばれるプログラム部品である。Java対応のWebブラウザでは、Webページ内にサーバー情報を伴ったアプレットを示す記述があると、リモートのサーバー計算機のファイルシステムにあるアプレットファイルをローカル計算機に転送し、ローカルマシン内のJava実行システムを用いて、アプレットを実行・表示する。このアプレット自身も、リモート計算機のファイルシステム内の別ファイルに実装された他のJavaプログラム部品を用いて構成することができ、アプレット実行時に、ローカル計算機のJava実行システムは、必要な部品のファイルをローカル計算機に新たに転送して実行する。

【0005】上記従来技術では、リモート計算機からローカル計算機に必要なファイルだけ転送するので、ファイルのトータルのサイズを一般に小さく保つことができ、一般に時間のかかるネットワークを介してファイルを転送する処理を無駄に行わずに済むという利点があった。

【0006】

【発明が解決しようとする課題】しかし、上記従来技術では、特に大きなサイズのファイルを転送する場合に、その間ユーザを待たせてしまうという問題があった。また、JAVAを用いても、必要なファイルが複数に及ぶ場合はそれらすべてをリモート計算機からローカル計算機に転送するので、結果的に大きなサイズのファイルを転送することになりユーザを待たせることがあった。

【0007】本発明の目的は、アプリケーションを構成するプログラム情報を格納する複数のファイルをリモート計算機からローカル計算機にダウンロードして実行するシステムにおいて、ファイルダウンロードにおけるユーザの待ち時間を削減することである。

【0008】

【課題を解決するための手段】上記目的を達成するため、請求項1に係る発明は、複数のファイルから構成されるアプリケーション・プログラムをファイルシステム上に保持するリモート計算機から、前記アプリケーション・プログラムを構成するファイル群をローカル計算機上のファイルシステムにダウンロードし、前記ローカル計算機上で前記ダウンロードしたファイルを内部記憶装置上に格納し、内部記憶装置上に格納したプログラムを

解釈して実行するアプリケーション実行方法において、アプリケーションを実行するのに必要なファイルがローカル計算機のファイルシステムに存在するか否かを判定する第1の処理ステップと、前記第1の処理ステップによりローカル計算機中のファイルシステムに前記ファイルが存在しないことが判明した場合、リモート計算機のファイルシステムからローカル計算機のファイルシステムに前記ファイルをダウンロードする第2の処理ステップと、前記ファイルをダウンロードしたことを管理する第3の処理ステップと、前記ファイルをローカル計算機の内部記憶装置に格納するとともに、前記ファイルを内部記憶装置に格納したことを管理する第4の処理ステップと、前記アプリケーションを構成する複数のファイルのダウンロード処理の優先順位を表す優先順位情報をアプリケーションを構成するファイルの1つの中に含め、前記アプリケーションの実行前に、前記優先順位情報を含むファイルをローカル計算機にダウンロードし、ダウンロードした前記優先順位情報を用いてアプリケーション構成ファイルのダウンロードの優先順位を記述した優先順位テーブルを作成する第5の処理ステップと、前記第1、第2、第3、および第4の処理ステップと並行して行われる処理であって、前記優先順位テーブルを参照し、その優先順位の順にリモート計算機上のファイルシステムからローカル計算機上のファイルシステムにファイルをダウンロードし、ダウンロードしたファイルを管理する第6の処理ステップとを備えたことを特徴とする。

【0009】請求項2に係る発明は、請求項1において、前記アプリケーション実行時に、実行対象のファイルを管理する処理ステップと、前記アプリケーションを構成するファイルの1つの中に、そのアプリケーションを構成する複数ファイル間の依存関係情報を含め、前記依存関係情報および前記実行対象ファイルの管理情報に基づいて、ダウンロードの優先順位をアプリケーションの実行時に動的に決定する処理ステップとをさらに備えたことを特徴とする。

【0010】請求項3に係る発明は、複数のファイルから構成されるアプリケーション・プログラムをファイルシステム上に保持するリモート計算機から、前記アプリケーション・プログラムを構成するファイル群をローカル計算機上のファイルシステムにダウンロードし、前記ローカル計算機上で前記ダウンロードしたファイルを内部記憶装置上に格納し、内部記憶装置上に格納したプログラムを解釈して実行するアプリケーション実行方法において、前記ローカル計算機上で、前記アプリケーションを構成するファイルのうちの1つを内部記憶装置上に格納して実行する実行ステップと、前記ローカル計算機上で、前記実行ステップとは別プロセスで並行して、前記アプリケーションを構成する他のファイルのダウンロードを順次行うダウンロードステップとを備えたことを

特徴とする。

【0011】

【発明の実施の形態】以下、図面を用いて本発明の実施の形態を説明する。

【0012】図1は、本発明に係るアプリケーション実行方法を実現するために用いる計算機の構成(100)の例を示すものである。前記計算機は、内部記憶装置であるメモリ(106)、ファイルシステムで管理されるファイルの実体を格納する外部記憶装置(103)、アプリケーションの実行処理等を行う中央処理装置(104)、ユーザからの入力出力を受けるための入出力装置(102)、計算機の処理内容等を表示する表示装置(101)、およびネットワークからのデータの入出力を行うネットワーク装置(105)から構成され、これらの構成要素は互いにデータを転送するためのバス(107)により接続されている。

【0013】図2は、本発明に係るアプリケーション実行方法を適用したアプリケーション実行システムの構成およびその動作環境(200)を示す。本アプリケーション実行システム(202)は、アプリケーション実行部(2021)、ダウンロード管理部(2022)、ロードモジュールファイル管理テーブル(2023)、ダウンロードファイル管理テーブル(2024)、ダウンロード情報(2026)、および優先順位テーブル(2025)を備えている。

【0014】次に、本アプリケーション実行システムの動作環境について述べる。動作環境としては、「従来の技術」で述べた従来技術であるWWWを利用する。計算機(201)は、アプリケーション実行システム(202)が実行される計算機であり、これをローカル計算機と呼ぶ。計算機(205)は、ネットワーク(204)を介してローカル計算機(201)と接続されている。前記計算機(205)をリモート計算機と呼ぶ。ローカル計算機(201)とリモート計算機(205)は図1に示した構成を有する。リモート計算機(205)上では、サーバプログラム(206)と呼ばれるプログラムが実行されており、公知のネットワーク技術を利用して、ローカル計算機(201)上で実行されているアプリケーション実行システム(202)が、リモート計算機(205)上のサーバプログラム(206)に対し

リモート計算機(205)上のファイルシステム(207)中にあるファイルの転送要求を行い、その要求に対して、リモート計算機(205)は要求対象となっているファイルをローカル計算機(201)に転送し、ローカル計算機(201)のファイルシステム(203)に格納する。

【0015】本実施形態では、一つのアプリケーションは、2つ以上の複数のファイルから構成される。これらのファイルをモジュールファイルと呼ぶ。モジュールファイルには、アプリケーションを構成するプログラムと

該プログラム実行に必要なデータが格納されている。特に、アプリケーションの実行開始時に実行されるプログラムを含むファイルを先頭モジュールファイルと呼ぶ。モジュールファイルは、ローカル計算機(201)のファイルシステム(203)、または、リモート計算機(205)のファイルシステム(207)に存在する。

【0016】本実施形態では、ローカル計算機(201)がアプリケーションの実行時に必要とするファイルをリモート計算機(205)に対して転送要求を行い、ローカル計算機(201)が前記ファイルを取得し、アプリケーションの実行を行う。

【0017】図3は、ロードモジュールファイル管理テーブル(2023)のデータ構造(300)を示すものである。ロードモジュールファイル管理テーブル(2023)は、メモリ上にロードされたモジュールファイルを管理するためのテーブルであり、ロードモジュールファイルエントリ(301)と呼ばれるデータ構造を要素として持つ配列である。ロードモジュールファイルエントリ(301)は、モジュールファイル名称と呼ばれる文字列(3011)、およびロードされたモジュールファイルのメモリ中の先頭アドレス(3012)から構成されるデータ構造である。

【0018】図4は、ダウンロードファイル管理テーブル(2024)のデータ構造(400)を示すものである。ダウンロードファイル管理テーブル(2024)は、ダウンロードされたモジュールファイルを管理するためのテーブルであり、ダウンロードファイルエントリ(401)と呼ばれるデータ構造を要素として持つ配列である。ダウンロードファイルエントリ(401)は、モジュールファイル名称と呼ばれる文字列(4011)、およびダウンロード状態(4012)から構成されるデータ構造である。ダウンロード状態(4012)は、整数型のデータ構造であり、対応するモジュールファイルの状態として、DOWNLOADEDまたはNOT_DOWNLOADの2つの値の何れかをとり。

【0019】図5は、ダウンロード情報(2026)のデータ構造(500)を示すものである。ダウンロード情報(2026)は、バックグラウンドでダウンロードを行う際(詳しくは後述するが、本アプリケーション実行システムでは、アプリケーションの実行処理を行うメイン処理プロセスのバックグラウンドでダウンロードを行うダウンロードプロセスを実行するようになっている)のスレッドの制御のために用いるテーブルである。ダウンロード情報(2026)は、スレッドフラグ(501)、カレントダウンロードファイルインデックス(502)、リモート計算機のネットワークアドレス(503)、モジュールファイルのリモート計算機上での格納先ディレクトリ(504)、およびインストールディレクトリ(505)からなる。

【0020】スレッドフラグ(501)は、整数型のデ

ータ型で、値としてAPPLICATION_QUIT、DOWNLOAD_QUIT、APPLICATION_RUNの3つの値をとる。スレッドフラグ(501)は、アプリケーションを実行するプロセスとダウンロードを実行するプロセスとの間で同期をとるために用いるフラグである。カレントダウンロードファイルインデックス(502)は、整数型のデータ構造であり、優先順位テーブル(600)中のあるエントリのインデックス値(どこまでダウンロードしたかを示すインデックス値)を保持する。リモート計算機のネットワークアドレス(503)は、アプリケーション実行システム(202)が実行の対象とするアプリケーションを構成するファイルをファイルシステム中に保持するリモート計算機(207)のネットワーク上のアドレスを保持するデータ構造である。モジュールファイルのリモート計算機上での格納先ディレクトリ(504)は、前記リモート計算機(207)のファイルシステム(207)に格納されているアプリケーションのモジュールファイルの格納先ディレクトリパスを保持するデータ構造である。インストールディレクトリ(505)は、モジュールファイルのローカル計算機(201)上での格納先ディレクトリである。

【0021】図6は、優先順位テーブル(2025)のデータ構造(600)を示すものである。優先順位テーブル(2025)は、1つのアプリケーション実行ごとに用意されるテーブルであり、そのアプリケーションを構成する複数のモジュールファイルをダウンロードする際の優先順位を格納したテーブルである。優先順位テーブル(2025)は、優先順位テーブルエントリ(601)と呼ぶデータ構造を要素として持つ配列である。優先順位テーブルエントリ(601)は、優先順位を表すインデックスと呼ぶ整数型のデータ構造(6011)およびモジュールファイル名称と呼ぶ文字列(6012)から構成されるデータ構造である。

【0022】図7は、本実施の形態で述べるアプリケーション実行システムが実行対象とするプログラムの構造(700)である。先に述べたようにアプリケーション(700)は、1つ以上の複数のモジュールファイルから構成される。先頭モジュールファイル(701)は、イベントマップ(702)、処理ブロックの配列(703)、プログラム実行に必要なデータ(704)、リモート計算機(205)のネットワーク上の位置情報(705)、モジュールファイルのリモート計算機上での格納先ディレクトリパス情報(706)、モジュールファイル優先順位情報(707)、およびアプリケーションのモジュールファイル構成(708)から構成される。モジュールファイル(709)は、処理ブロックの配列(710)およびアプリケーションの実行に必要なデータ(711)から構成される。

【0023】処理ブロックは、プログラムを構成する手

続きであり、アプリケーション実行部(2021)により解釈実行可能な形態で、ファイルに格納されたものである。リモート計算機のネットワーク上の位置情報(705)は、リモート計算機(205)のネットワーク上の位置情報を保持する領域である。モジュールファイルのリモート計算機(205)上での格納先ディレクトリパス情報(706)は、アプリケーション実行システム(202)の実行対象となるアプリケーションを構成するモジュールファイルのリモート計算機(205)のファイルシステム(207)上の格納先ディレクトリパス情報を保持する領域である。モジュールファイル構成(708)は、このアプリケーションを構成するモジュールファイル名称の配列である。

【0024】図8は、モジュールファイル優先順位情報(707)のデータ構造(800)を示す図である。モジュールファイル優先順位情報(800)は、モジュールファイルをリモート計算機(205)からダウンロードする際の優先順位情報を保持する領域であり、優先順位エントリ(801)の配列である。優先順位エントリ(801)は、整数型の優先順位(8011)およびモジュールファイル名称(8012)である文字列から構成される。優先順位(8011)は、1から順にモジュールファイル数だけ割り振られ、同じ優先順位を持つモジュールファイルはない。

【0025】次にイベントマップについて説明する。本実施形態のアプリケーション実行部(2021)のアプリケーションの実行方式は、イベント駆動型と呼ばれるものであるとする。イベント駆動型のアプリケーション実行部(2021)は、以下のデータを管理する。すなわち、イベントキュー、およびイベントマップの先頭アドレスである。

【0026】イベントとは、数値型のデータである。イベントキューは、前記イベントのリストであり、2つの関数、すなわちPostEvent関数およびGetEvent関数により、操作されるものとする。PostEvent関数は、イベントを引数としてもつ関数であり、イベントキューの最後に引数イベントを加える処理を行い、戻り値はない。入出力装置(102)で発生する入出力イベントはオペレーティング・システムからPostEvent関数を通して、アプリケーションに通知される。GetEvent関数は、引数なしの関数であり、戻り値として、イベントキューの先頭のイベントを返し、イベントキューの先頭イベントを削除する。

【0027】イベントキューが長さ0のとき、GetEvent関数は、制御を呼び出し側に戻さず、呼び出し側のスレッドの実行をアイドル状態にする。別スレッドあるいはオペレーティング・システムが、前記イベントキューに対して、PostEventを実行し、イベントキューが長さ1以上になった時点で、前記GetEvent関数の呼び出しのアイドル状態を開放し、実行を

再開するものとする。

【0028】イベントマップの先頭アドレスは、イベントマップ（図9で詳述する）と呼ぶデータのメモリ中の先頭アドレスを保持するデータ構造である。

【0029】図9は、イベントマップ（702）のデータ構造（900）を示したものである。イベントマップ（900）は、イベント管理情報（901）と呼ばれるデータ構造を要素として持つ配列である。イベント管理情報（901）は、イベント（9011）、当該イベントに対応する処理ブロックのモジュール内での相対アドレス（9012）、当該処理ブロックを含むモジュールファイル名称（9013）からなるデータ構造である。

【0030】イベントマップ（900）は、DispatchMap関数を通して操作（参照）される。DispatchMap関数は、引数としてイベントをとり、「イベントマップの先頭アドレス」で指定されるイベントマップ（900）を走査して、引数で指定されたイベントに対応する処理ブロックのアドレス（9012）と当該処理ブロックを含むモジュール名称（9013）の対をDispatchMap関数の戻り値として返す。

【0031】アプリケーションの特別なイベントとして、QUITイベントとSTARTイベントがある。STARTイベントに対してアプリケーションの実行開始処理ブロックが、QUITイベントに対してアプリケーションの終了処理ブロックが、それぞれイベントマップ（900）中で対応付けられる。イベント駆動型のアプリケーションの実行の終了は、プログラム実行中にQUITイベントが発生することにより行われる。

【0032】以上が、本実施形態で述べるプログラムの構成の一例である。これはあくまで、以降で述べる本発明に係るプログラム実行方法の実施形態を具体的に記述するために便宜的に定義したものであることをここで断っておく。

【0033】次に、先頭モジュールファイル（701）のダウンロードとアプリケーション実行システム（202）の起動について、図10および図20を用いて述べる。本実施形態においては、先頭モジュールファイル（701）のダウンロードと、これまで説明したアプリケーション実行システム（202）の起動は、「従来の技術」で述べた通り、従来技術であるWWWとWebブラウザ（1002）の機能を用いて行われる。

【0034】図20は、HTMLのスク립ト例（2000）である。図10において、Webブラウザを構成する処理（10021）～（10025）および（1002）は、Webブラウザを説明するために便宜的にここで定義したものであり、実際のWebブラウザのシステム構成と必ずしも一致するものではないことを断っておく。Webブラウザ（1002）は、HTML表示処理（10022）により、HTML（Hyper Text Markup Language）スク립トを

読み込んでブラウザ画面（1001）に文字、図形、画像等を表示する。HTMLの詳細については、例えば、文献：“OpenDesign, No. 13, 1996年4月、CQ出版社、p4からp13、p40からp59”で説明されている。

【0035】図20のHTMLスク립ト（2000）において、文字、図形、画像等に対してハイパーリンク先であるURL情報が対応づけられているとき、これをアンカーと呼ぶ。HTMLスク립ト（2000）の例では、（2001）に示すように、図10のブラウザ画面（1001）上の文字“アプリケーション1”はアンカーであり、URL“http://zzz.hitachi.co.jp/App1/apl1.fst”が対応づけられている。apl1.fstが、アプリケーション実行システム（202）が実行対象とするアプリケーションの先頭モジュールファイルのファイル名称である。

【0036】入力処理（10021）により、ブラウザ画面（1001）上で文字列“アプリケーション1”が入出力装置（102）を通してクリックされたことが判明すると、ファイル転送要求処理（10023）は、URL情報を用いて、本例ではネットワーク上の位置情報が“zzz.hitachi.co.jp”であるリモート計算機（205）に対して、通信プロトコル“HTTP”で、ファイル“App1/apl1.fst”のファイル転送を要求する。前記転送要求に対して、リモート計算機（205）上のサーバプログラム（206）は、要求された前記ファイル“apl1.fst”をローカル計算機（201）のファイルシステム（203）に転送する。本実施形態においては、先頭モジュールファイル（701）はこのようにして、ローカル計算機（201）のファイルシステム（203）に転送される。さらに、Webブラウザ（1002）の結果解析処理（10024）は、ダウンロードされたファイルのヘッダ情報を解析し、転送されたファイルの種別を特定し、その後の処理を振り分ける。例えば、ファイルの種別がHTMLファイルである場合には、HTML表示処理（10022）によりWebブラウザ画面（1001）に表示を行わせる。あらかじめ、ブラウザプロセス（1002）は、ファイル種別とそのファイルを処理することができるアプリケーションを対応付けた「AP起動対応表」（10025）を持っている。結果解析処理（10024）は、ファイルダウンロード後、このAP起動表（10025）を参照して、転送されたファイルのファイル種別に対応するアプリケーションを起動する。本実施形態では、先頭モジュールファイル（701）のダウンロード後、そのダウンロードした前記先頭モジュールを入力として、アプリケーション実行部（2021）のアプリケーションの実行処理を開始する。

【0037】ローカル計算機（201）のアプリケーシ

ョン実行部(2021)のアプリケーションの実行処理(1100)とダウンロード管理部(2022)の実行(1700)とは、別スレッドとして、並行して実行される。本発明の実施形態は、マルチスレッドが可能で、かつスレッドの優先順位の指定ができるオペレーティング・システム上で実現されるものとする。

【0038】また、アプリケーションの実行処理(1100)のスレッド(メイン処理スレッドと呼ぶ)とダウンロード管理部の実行(1700)のスレッド(ダウンロードスレッドと呼ぶ)とは、アプリケーションの実行の終了のタイミングにおいて、2つのスレッドの終了処理に関して同期をとる必要がある。そこで、前記2つのスレッド間で、前記スレッドフラグ(501)と呼ぶデータを通して、同期をとる。

【0039】図11は、アプリケーション実行部(2021)によるアプリケーションの実行処理(1100)のフローチャートを示している。本処理は、アプリケーション実行部(2021)の起動時の入力である先頭モジュールファイルを入力として持ち、本処理内のローカル変数として、変数EVENT、変数PROC、および変数MODULを持つ。

【0040】まずステップ(1101)で、アプリケーション実行部(2021)の実行開始処理(図12の1200)を行う。次にステップ(1102)で、STARTイベントを引数としてPostEventを実行し、当該アプリケーションのイベントキューの先頭にSTARTイベントを挿入する。ステップ(1103)で、GetEvent関数の実行により、イベントキューの先頭イベントを取得し、変数EVENTに格納する。判断(1104)で、変数EVENTの値がQUITイベントであるか判定する。この判定結果が真である場合、ステップ(1107)に飛ぶ。この判定結果が偽である場合は、ステップ(1105)に飛ぶ。ステップ(1105)では、変数EVENTの値を引数として、DispatchMap関数を実行し、変数EVENTの値に対応する処理ブロックのモジュール内の相対アドレスと、当該処理ブロック含むモジュールファイル名を取得する。前記処理ブロックのアドレスを変数PROCに、モジュールファイル名を変数MODULに格納する。次にステップ(1106)で、変数PROCの値と変数MODULの値を入力として、アプリケーション実行部(2021)のプログラム処理ブロックの実行処理(図14の1400)を実行する。前記処理ブロックの実行の終了後、ステップ(1103)に戻る。

【0041】ステップ(1107)では、アプリケーションの終了処理(1500)を行う。ステップ(1107)の後、本処理を終了する。

【0042】図12は、ステップ(1101)で実行するアプリケーション実行部(2021)の実行開始処理(1200)のフローチャートを示している。この処理

は、先頭モジュールファイルを入力として持つ。本処理内のローカル変数としてnを持つ。

【0043】ステップ(1201)で、本アプリケーション実行のイベントキューを生成する。この時点でイベントキューの長さは0である。ステップ(1202)で、入力である先頭モジュールファイル(701)をロードする。ロードされた先頭モジュールファイル(701)のイベントマップ(702)のメモリ上のアドレスをイベントマップの先頭アドレスに設定する。これにより、先頭モジュールファイル(701)中に含まれるイベントマップ(702)が、関数DispatchMapを通してアクセス可能となる。

【0044】次に、ステップ(1203)で、ロードされた先頭モジュールファイル(701)のサーバー位置情報(705)、アプリケーションのリモート計算機上での格納先ディレクトリパス(706)を、それぞれ、ダウンロード情報(2026、図5の500)のリモート計算機のネットワーク位置情報(503)、アプリケーションのリモート計算機上での格納先ディレクトリパス(504)に格納する。また、ダウンロード情報(500)のカレントダウンロードファイルインデックス(502)に1を設定する。さらにダウンロード情報(500)のスレッドフラグ(501)をAPPLICATION_RUNに設定する。ダイアログボックスを表示し、ユーザにインストールディレクトリパス(505)の値を設定してもらう。次に、アプリケーションモジュールファイル構成(708)の長さで、ダウンロードファイル管理テーブル(2024、図4の400)、ロードモジュールファイル管理テーブル(2023、図3の300)を初期化する。具体的には、まず、各テーブルの確保した配列領域(当該アプリケーションを構成するファイルモジュールの数分のデータエントリが確保される)を0で初期化する。ダウンロードファイル管理テーブル(400)の1番目の要素のモジュールファイル名称(4011)に、先頭モジュールファイル(701)のファイル名称を設定する。ロードモジュールファイル管理テーブル(300)の1番目の要素のモジュールファイル名称(3011)に、先頭モジュールファイル(701)のファイル名称を設定する。ダウンロードファイル管理テーブル(400)の1番目の要素のダウンロード状態(4012)には、DOWNLOADEDを設定する。ロードモジュールファイル管理テーブル(300)の1番目の要素の先頭アドレス(3012)には、ロードされた先頭モジュールファイル(701)の先頭アドレスを設定する。ローカル変数nに2を設定する。以上のステップ(1203)の処理の後、ステップ(1204)に進む。

【0045】判断(1204)で、式「n<=モジュールファイル構成(708)の長さ」を評価し、その結果が真である場合は、ステップ(1205)に飛ぶ。前記

評価結果が偽の場合は、ステップ(1208)に飛ぶ。ステップ(1205)では、ロードモジュールファイル管理テーブル(300)のn番目の要素のモジュールファイル名称(3011)に、先頭モジュールファイル(701)のモジュールファイル構成(708)のn番目の要素のモジュールファイル名称を設定する。ステップ(1206)では、ダウンロードファイル管理テーブル(400)のn番目の要素のモジュールファイル名称(4011)に、先頭モジュールファイル(701)のモジュールファイル構成(708)のn番目の要素のモジュールファイル名称を設定する。ステップ(1207)で、式「 $n := n + 1$ 」を実行する。ステップ(1208)では、優先順位テーブルの生成処理(図13の1300)を行い、後述する処理(図17の1700)を行うダウンロードスレッドを生成する。

【0046】図13は、ステップ(1208)で実行する優先順位テーブルの生成処理(1300)のフローチャートを示す図である。本処理は、入力として先頭モジュールファイルを持ち、本処理内のローカル変数としてnを持つ。ステップ(1301)で、優先順位テーブル(2025、図6の600)を、入力である先頭モジュールファイル(701)のモジュールファイル構成(708)の長さで初期化する。次にnに1を設定する。判断(1302)で、式「 $n \leq \text{モジュールファイル構成(708)の長さ}$ 」を評価し、その評価結果が真である場合、ステップ(1303)に飛ぶ。前記評価結果が偽である場合、本処理を終了する。ステップ(1303)で、優先順位テーブル(600)のn番目の要素のインデックス(6011)に、優先順位情報(707)のn番目の要素の優先順位(8011)を設定する。ステップ(1304)で、優先順位テーブル(600)のn番目の要素のモジュールファイル名称(6012)に、優先順位情報(707)のn番目の要素のモジュールファイル名称(8012)を設定する。ステップ(1305)で、式「 $n = n + 1$ 」を実行する。ステップ(1305)の後、ステップ(1302)へ戻る。

【0047】図14は、ステップ(1106)のアプリケーション実行部(2021)によるプログラム処理ブロックの実行処理(1400)のフローチャートを示すものである。本処理は、実行対象となる処理ブロックのモジュール内の相対アドレスと前記処理ブロックを含むモジュール名を入力として持ち、本処理内のローカル変数として、変数MODULTOPを持つ。

【0048】ステップ(1401)で、入力であるモジュール名を入力としてダウンロード処理(図19の1900)を実行する。なお、ここでいうダウンロード処理はダウンロード命令の発行を行うものである(実際のダウンロードはダウンロードスレッドで実行する)。ステップ(1402)で、入力であるモジュールファイル名称を入力として、アプリケーション実行部(2021)

によるロード処理(図15の1500)を行う。前記処理のリターン値を変数MODULTOPに格納する。ステップ(1403)で、変数MODULTOPには、入力であるモジュールファイルのメモリ中の先頭アドレスが格納されている。前記アドレスをベースアドレス、入力である処理ブロックのアドレスをオフセットアドレスとして、メモリ中の処理ブロックの実アドレスを計算し、アプリケーション実行部(2021)が処理ブロックを解釈・実行する。以上で本処理を終了する。

【0049】図15は、ステップ(1402)のアプリケーション実行部(2021)によるモジュールファイルのロード処理(1500)のフローチャートを示すものである。本処理は、入力としてモジュールファイル名称を持ち、出力としてメモリのアドレスを返す。また、本処理は、本処理内のローカル変数として、変数MODULTOPを持つ。

【0050】判断(1501)で、入力であるモジュールファイルがすでにダウンロードされているかを調べる。これは、ダウンロードファイル管理テーブル(2024、図4の400)を走査し、入力であるモジュールファイルに対応するダウンロードファイルエントリ(401)のダウンロード状態(4012)がDOWNLOADEDと等しくなっているかを判定することにより行う。もし、前記判定で真となっている場合、入力であるモジュールファイルはダウンロードされている。前記判定で偽となっている場合、入力であるモジュールファイルはダウンロードされていない。ダウンロードされている場合は、ステップ(1502)に飛ぶ。ダウンロードされていない場合は、ステップ(1503)に飛ぶ。ステップ(1503)では、入力であるモジュールファイル名称を入力としてダウンロード処理(図19の1900)を実行する。ステップ(1503)の後、ステップ(1504)に進む。

【0051】判断(1502)では、入力であるモジュールファイルがメモリ中にロードされているか判定する。この判定は、ロードモジュールファイル管理テーブル(2023、図3の300)を走査し、入力であるモジュールファイルに対応するロードモジュールファイルエントリ(301)の先頭アドレス(3012)が0以外になっている場合、前記判定結果を真とし、それ以外を偽とする。この判定結果が真の場合、変数MODULTOPにロードモジュールファイルエントリ(301)の先頭アドレス(3012)を設定して、ステップ(1505)に飛ぶ。ステップ(1502)の判定結果が偽である場合は、ステップ(1504)に飛ぶ。

【0052】ステップ(1504)では、入力であるモジュールファイルをメモリ中にロードし、その先頭アドレスを変数MODULTOPに格納する。ロードモジュールファイル管理テーブル(2023、図3の300)を走査し、入力であるモジュールファイル名称に対応す

るロードモジュールファイルエントリ(301)を取得する。前記ロードモジュールファイルエントリ(301)の先頭アドレス(3012)に変数MODULTOPの値を格納する。ステップ(1505)では、本処理の出力として、変数MODULTOPの値を返す。以上で本処理を終了する。

【0053】図16は、ステップ(1107)のアプリケーション実行部(2021)によるアプリケーション終了処理(1600)のフローチャートである。判断(1601)で、スレッドフラグ(501)がDOWNLOAD_QUITになっているかを判定する。この結果が真である場合は、ステップ(1604)に飛ぶ。前記判定の結果が偽である場合は、ステップ(1602)に飛ぶ。ステップ(1602)で、スレッドフラグ(501)をAPPLICATION_QUITに設定する。ステップ(1603)で、ダウンロードスレッドが終了するまで、メイン処理のスレッドを「アイドル状態」にし、ダウンロードスレッドの終了後、以降の処理が再開されるようにスレッド制御を行うようオペレーティング・システムに要求する。ステップ(1604)で、アプリケーションの実行を終了する。以上で本処理を終了する。

【0054】図17は、第二のスレッドであるダウンロードスレッド(図12のステップ(1208)で生成)で実行される処理(1700)のフローチャートである。ダウンロードスレッドで行われる処理は、メイン処理スレッドと並行して行われる。ただし、ダウンロードスレッドは、メイン処理スレッドよりスレッドの優先順位を低く設定して実行される。本処理は、入力はないが、本処理のローカル変数としてFILEを持つ。

【0055】判断(1701)で、スレッドフラグ(501)にAPPLICATION_QUITが設定されているか否かを判定する。前記判定の結果が真である場合、ステップ(1705)に飛ぶ。前記判定の結果が偽である場合、判断(1702)に飛ぶ。判断(1702)で、すべてのファイルがダウンロードされているか否かの判定を行う。この判定では、ダウンロードファイル管理テーブル(400)中の各エントリ(401)を走査し、すべてのエントリ(401)のダウンロード状態(4012)がDOWNLOADEDになっている場合、前記判定結果を真とし、それ以外の場合、偽とする。本判断の結果が真の場合は、ステップ(1705)に飛ぶ。本判断の結果が偽の場合、ステップ(1703)に飛ぶ。ステップ(1703)で、ダウンロードファイル決定処理(図18の1800)を実行する。その結果を変数FILEに格納する。ステップ(1704)で、変数FILEの値を入力として、ダウンロード処理(図19の1900)を実行する。ステップ(1704)の後、ステップ(1701)に戻る。ステップ(1705)では、スレッドフラグ(501)をDOWNL

OAD_QUITに設定し、ダウンロードスレッドを終了する。

【0056】図18は、ダウンロードスレッドのステップ(1703)で実行されるダウンロードファイル決定処理(1800)のフローチャートである。本処理は、入力はないが、出力としてダウンロードすべきモジュールファイル名を返す。また、本処理は、本処理内のローカル変数としてFILEを持つ。

【0057】ステップ(1801)で、変数FILEに空文字列を設定する。ステップ(1802)で、カレントダウンロードファイルインデックス(502)の値を1だけインクリメントする。判断(1803)で、カレントダウンロードファイルインデックス(502)に対応するインデックス値を持つ優先順位テーブル(600)中のテーブルエントリ(601)が存在するか否かの判定を行う。前記判定の結果が真(存在する)である場合、ステップ(1804)に飛ぶ。前記判定が偽である場合は、ステップ(1811)に飛ぶ。

【0058】ステップ(1804)で、カレントダウンロードファイルインデックス(502)に対応する優先順位テーブル(600)中のテーブルエントリ(601)を取得する。ステップ(1805)で、前記テーブルエントリ(601)のモジュールファイルのダウンロード状態(4012)を取得する。判断(1806)で、前記ダウンロード状態がDOWNLOADEDであるか否かを判定する。この判定結果が真である場合、ステップ(1802)に飛ぶ。前記判定結果が偽である場合、ステップ(1807)に飛ぶ。ステップ(1807)で、前記エントリのモジュールファイル名称の項目を参照し、その値を変数FILEに設定する。ステップ(1808)で、変数FILEの値を本処理の出力として返し、本処理を終了する。

【0059】図19は、ダウンロード処理(1900)のフローチャートである。本処理は、入力としてモジュールファイル名称をとる。ダウンロード処理(1900)は、図14のステップ(1401)、図15のステップ(1503)、およびダウンロードスレッドのステップ(1704)で実行される処理である。

【0060】判断(1901)で、入力であるモジュール名のファイルがローカルファイルシステム(203)中のインストールディレクトリパス(505)下にあるか検索し、存在する場合は、ステップ(1904)に飛ぶ。もし存在しなければ、ステップ(1902)に飛ぶ。ステップ(1902)で、リモート計算機のネットワーク上の位置情報(503)で特定されるリモート計算機(205)に対して、モジュールファイルのリモート計算機上での格納先ディレクトリ(504)で指定されるリモート計算機のファイルシステム上のディレクトリ下にあるファイルで、入力であるモジュールファイル名称と同じファイル名称を持つファイルの転送要求を行

う。ステップ(1903)で、転送されたファイルを、入力であるモジュールファイル名称と同じファイル名称で、インストールディレクトリパス(505)で示されるローカル計算機のファイルシステムのディレクトリの下に格納する。ステップ(1904)で、ダウンロードファイル管理テーブル(2024、図4の400)から、入力であるモジュールファイル名称に対応するテーブルエントリ(401)を取得し、そのテーブルエントリ(401)のダウンロード状態(4012)をDOWNLOADEDに設定する。

【0061】以上で、本発明の一実施形態であるアプリケーション実行システム(202)を説明した。上記第1の実施形態によれば、別スレッドでもモジュールファイルのダウンロードを行うので、ユーザとの対話的操作を多く含むアプリケーションファイルをダウンロードして実行する場合、ユーザ入力の待ち時間を利用してダウンロードを行うことができ、必要になったときには既にダウンロードが終了しているというようにできる。したがって、ファイルダウンロード時のユーザの待ち時間を削減してユーザの快適さを向上させることができる。

【0062】次に、先に述べたアプリケーション実行システム(202)を拡張した第2の実施の形態について述べる。上記第1の実施の形態では、ダウンロードする順序はスタティックに決定されていた。第2の実施の形態では、メモリに最新にロードされたモジュールファイルに依存するモジュールファイルを、次に実行される可能性の高いモジュールとして、早めにダウンロードするようにした。第2の実施の形態の構成や処理は、第1の実施の形態と同様であるので、以下では異なる部分のみ説明する。特に、先頭モジュールファイル(701)、ダウンロード情報のデータ構造(500)、処理(1500)、処理(1800)の変更点および新規に追加したデータ構造(2100)の説明を行う。

【0063】まず、先頭モジュールファイル(701)に、モジュールファイル間依存関係情報を付加する。モジュールファイル間依存関係情報については、図21で詳述する。

【0064】図21に、モジュールファイル間依存関係情報のデータ構造(2100)を示す。モジュールファイル間依存関係情報は、依存情報エントリ(2101)と呼ぶデータ構造の配列である。依存情報エントリ(2101)は、モジュールファイル名称(21011)および依存する他のモジュールファイル名称配列(21012)から構成される。依存する他のモジュールファイル名称配列(21012)は、モジュールファイル名称(21011)で特定されるモジュールファイルとプログラムの制御上依存関係にあるモジュールファイルのファイル名称の配列である。図21は、先頭モジュールファイル(701)に付加したモジュールファイル間依存

関係情報のデータ構造を示すが、同じデータ構造のテーブルをアプリケーション実行システム(202)中に用意し、図12の実行開始処理で先頭モジュールファイル(701)のモジュールファイル間依存関係情報(2100)を当該テーブルにコピーするステップを加える。これにより、後述する図22の処理でモジュールファイル間依存関係情報(2100)が参照できるようになる。

【0065】ダウンロード情報のデータ構造(500)に、リーセントロードモジュールファイル名称と呼ぶ文字列型のデータ構造を加える。処理(1200)のステップ(1203)と判断(1204)の間に、「リーセントロードモジュールファイル名称として先頭モジュールファイル名称を設定する」という処理を行うステップを挿入する。

【0066】さらに処理(1500)の「スタート」とステップ(1501)の間に、「リーセントロードモジュールファイル名称として、入力であるモジュールファイル名称を設定する」という処理を行うステップを挿入する。これにより、一番最近にメモリ上にロードされたモジュールファイルの名称がリーセントロードモジュールファイル名称として設定される。

【0067】ダウンロードファイル決定処理(1800)を、下記のように変更する。ステップ(1801)とステップ(1802)の間に、図22に示すフロー(2200)を挿入する。本フローは、ローカル変数n、およびローカル配列変数Xを持つ。

【0068】ステップ(2201)で、配列変数Xに、モジュールファイル間依存関係情報中の、リーセントモジュールファイル名に対応する依存関係情報エントリのモジュールファイル名称配列(21012)を設定する。ステップ(2202)で、式「 $n := 1$ 」を実行する。判定(2203)で、式「 $n \leq X$ の長さ」を評価する。評価結果が真であるとき、ステップ2204に飛ぶ。評価結果が偽である場合は、ステップ1802に飛ぶ。判定(2204)で、Xのn番目のファイル名称のファイルがローカル計算機のファイルシステムのインストールディレクトリ(505)下にあるか判定する。その判定結果が真である場合は、ステップ(2206)に飛ぶ。前記判定結果が偽である場合は、ステップ(2205)に飛ぶ。ステップ(2205)で、式「 $n := n + 1$ 」を実行し、ステップ(2203)に飛ぶ。ステップ(2206)で、変数FILEに、Xのn番目のファイル名称を設定し、ステップ(1808)に飛ぶ。

【0069】上記図22の処理により、次にダウンロードするモジュールファイルを決定する際に、最近ロードしたリーセントロードモジュールファイル名称のモジュールファイルに依存するモジュールファイルでダウンロードされていないものがあつたら、それを優先してダウンロードすることになる。以上で、第1の実施形態の拡

張である第2の実施形態の説明を終了する。

【0070】

【発明の効果】請求項1および3に係る発明によれば、アプリケーションを構成する複数のファイルをリモート計算機のファイルシステムからローカル計算機のファイルシステムにダウンロードして実行するシステム、特に、入出力装置を用いたユーザとの対話的操作を多く含むアプリケーションファイルをダウンロードして実行するシステムにおいて、ファイルダウンロード時のユーザの待ち時間を削減してユーザの快適さを向上させることができる。特に、請求項2のように動的にダウンロードするファイルを決断するようにすれば、よりユーザの待ち時間を削減してユーザの快適さを向上させることができる。

【図面の簡単な説明】

【図1】 本発明の実施の形態で用いる計算機の構成例を示す図

【図2】 アプリケーションシステムの構成およびその動作環境を示す図

【図3】 ロードモジュールファイル管理テーブル（2023）のデータ構造を示す図

【図4】 ダウンロードファイル管理テーブル（2024）のデータ構造を示す図

【図5】 ダウンロード情報（2026）のデータ構造を示す図

【図6】 優先順位テーブル（2025）のデータ構造を示す図

【図7】 本実施形態で述べるアプリケーション実行システムが実行対象とするプログラムの構造を示す図

【図8】 モジュールファイル優先順位情報（708）のデータ構造を示す図

【図9】 イベントマップ（702）のデータ構造を示す図

【図10】 先頭モジュールファイルのダウンロードとアプリケーション実行部の起動を説明する図

【図11】 アプリケーション実行部によるアプリケー

ションの実行処理のフローチャート図

【図12】 アプリケーション実行部の実行開始処理のフローチャート図

【図13】 優先順位テーブルの生成処理のフローチャート図

【図14】 アプリケーション実行部のプログラム処理ブロックの実行処理のフローチャート図

【図15】 アプリケーション実行部のモジュールファイルのロード処理のフローチャート図

【図16】 アプリケーション実行部のアプリケーション終了処理のフローチャート図

【図17】 ダウンロードスレッドで実行される処理のフローチャート図

【図18】 ダウンロードファイル決定処理のフローチャート図

【図19】 ダウンロード処理のフローチャート図

【図20】 HTMLのスクリプトの例を示す図

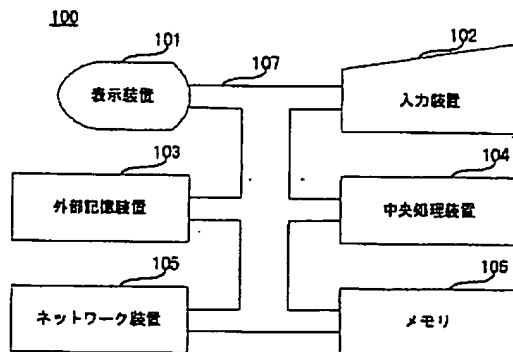
【図21】 モジュールファイル間依存関係情報のデータ構造を示す図

【図22】 ダウンロードファイル決定処理（1800）への追加フローチャート図

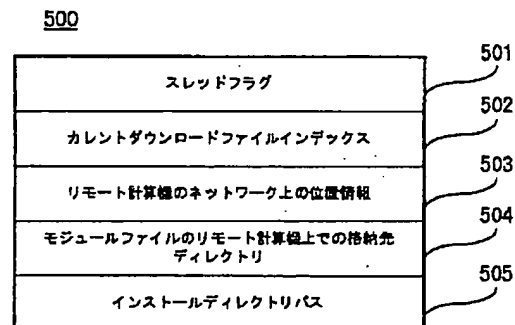
【符号の説明】

100…計算機、101…表示装置、102…入出力装置、103…外部記憶装置、104…中央処理装置、105…ネットワーク装置、106…メモリ、107…バス、200…アプリケーション実行システムの構成およびその動作環境、201…ローカル計算機、202…アプリケーション実行システム、203…ファイルシステム、204…ネットワーク、205…リモート計算機、206…サーバープログラム、207…ファイルシステム、2021…アプリケーション実行部、2022…ダウンロード管理部、2023…ロードモジュールファイル管理テーブル、2024…ダウンロードファイル管理テーブル、2026…ダウンロード情報、2025…優先順位テーブル。

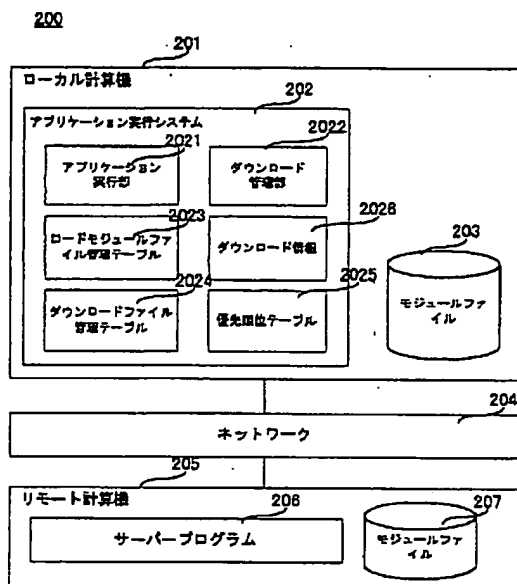
【図1】



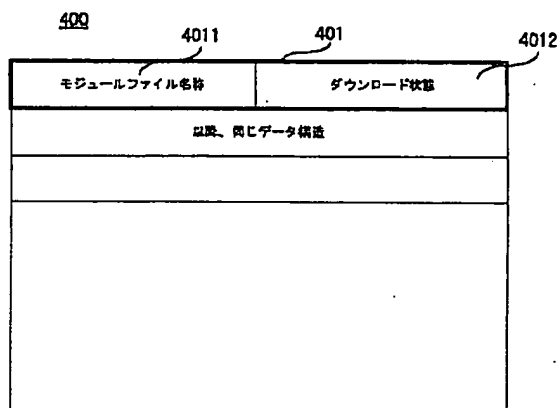
【図5】



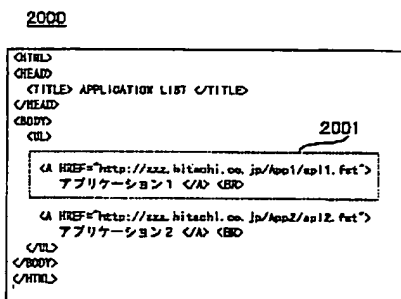
【図2】



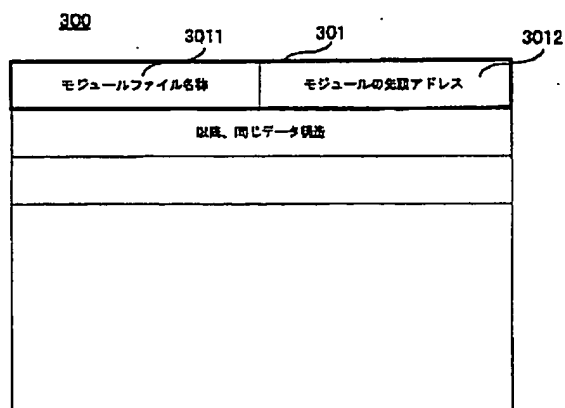
【図4】



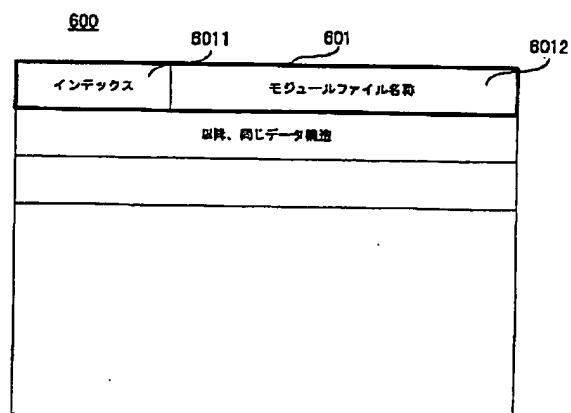
【図20】



【図3】

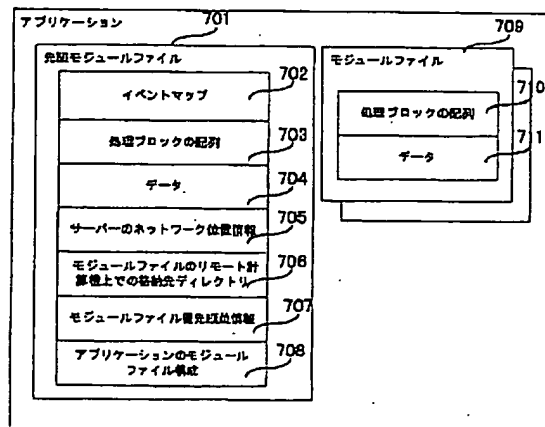


【図6】

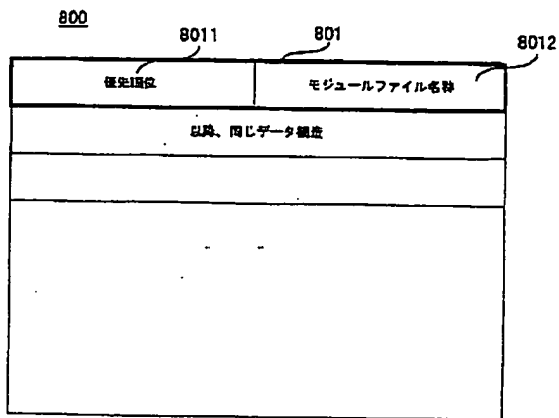


【図7】

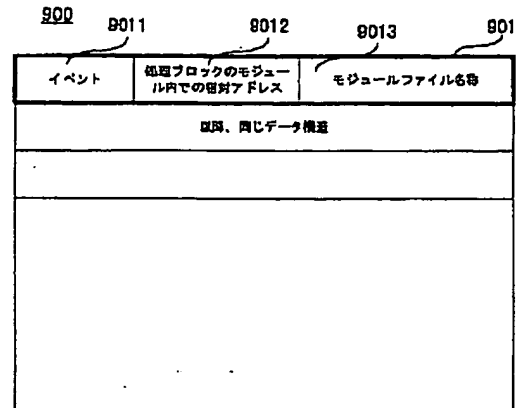
700



【図8】

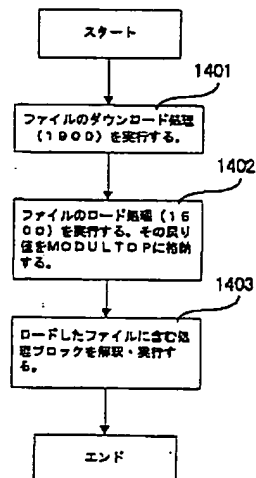


【図9】

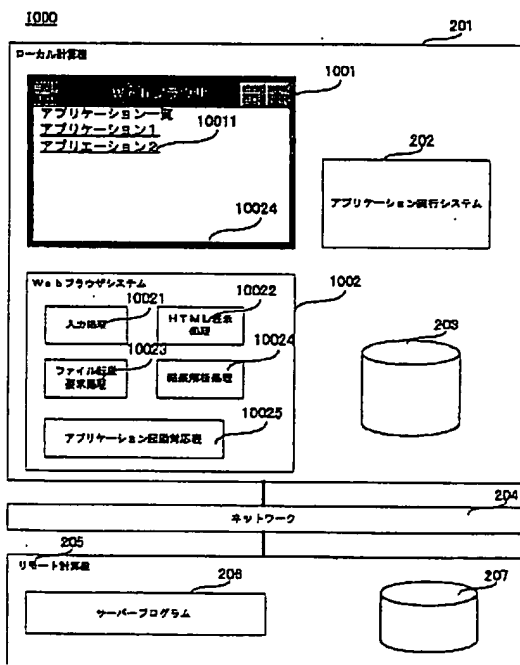


【図14】

1400

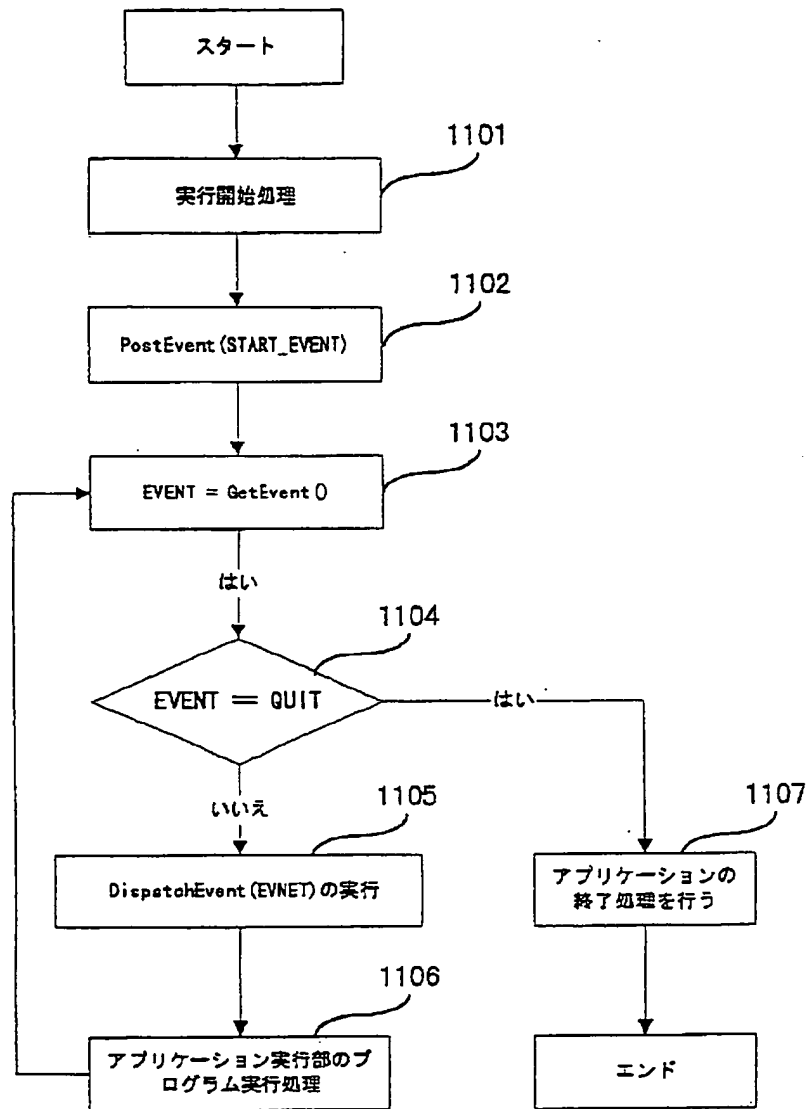


【図10】



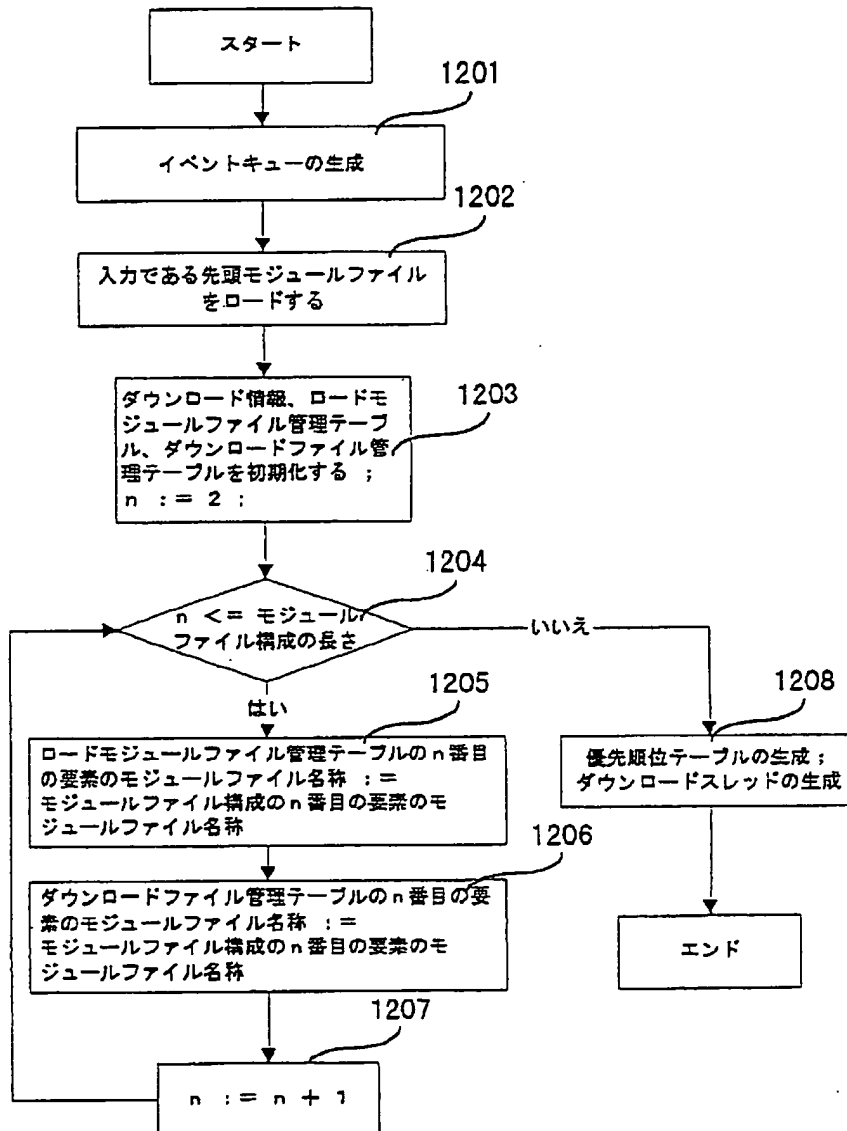
【図11】

1100

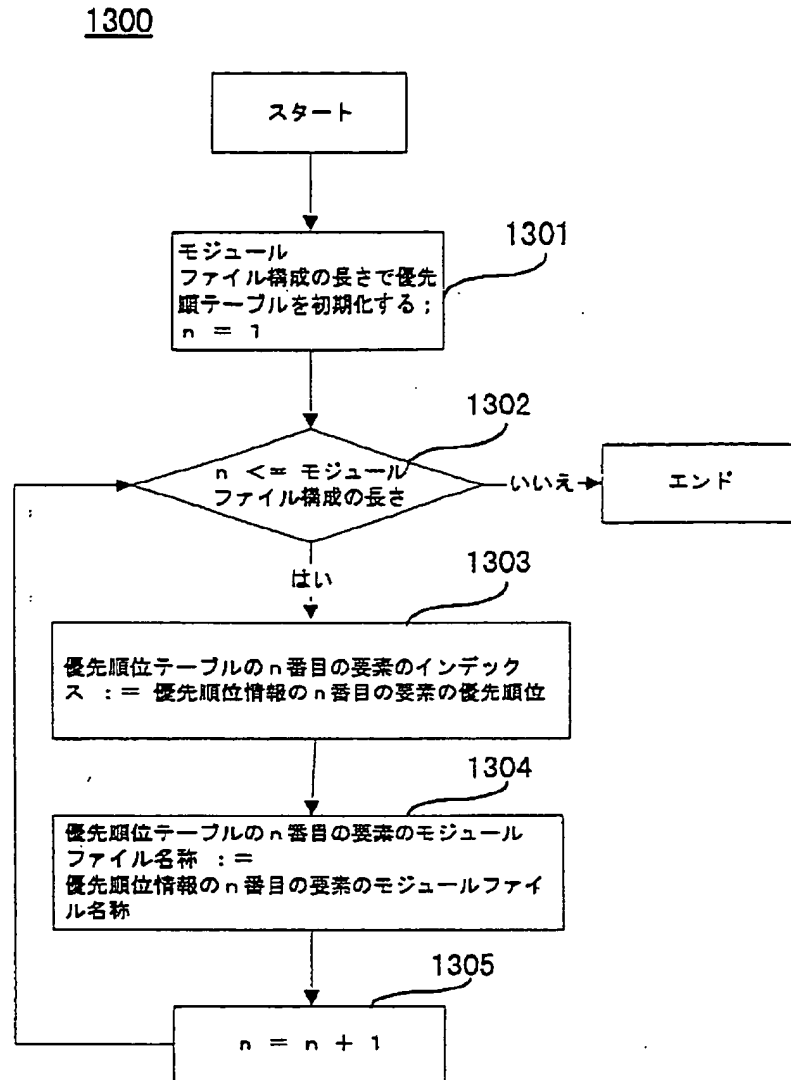


【図12】

1200

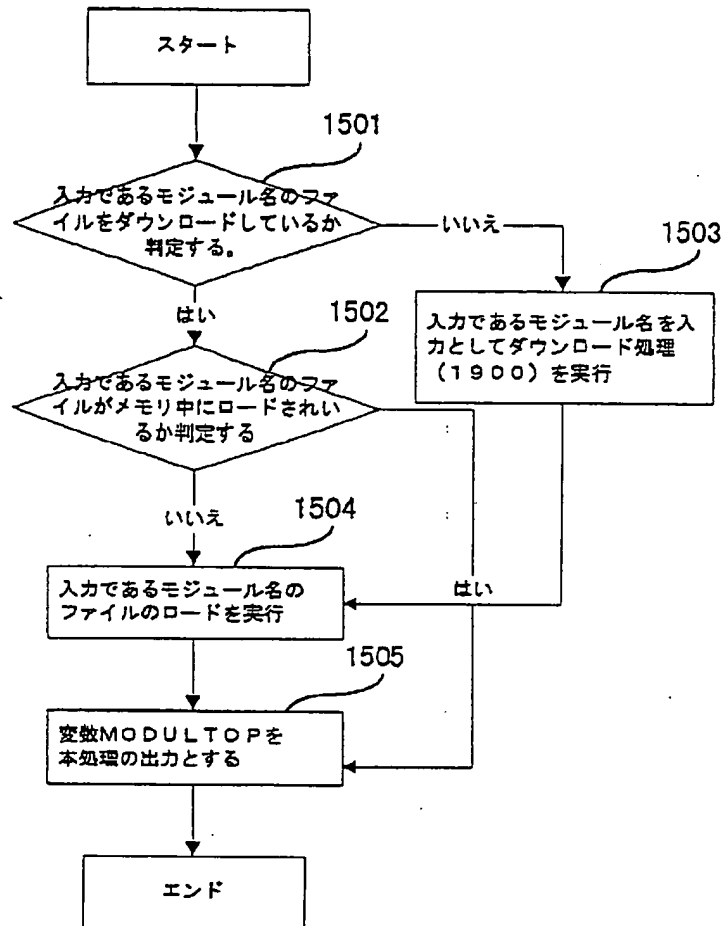


【図13】



【図15】

1500

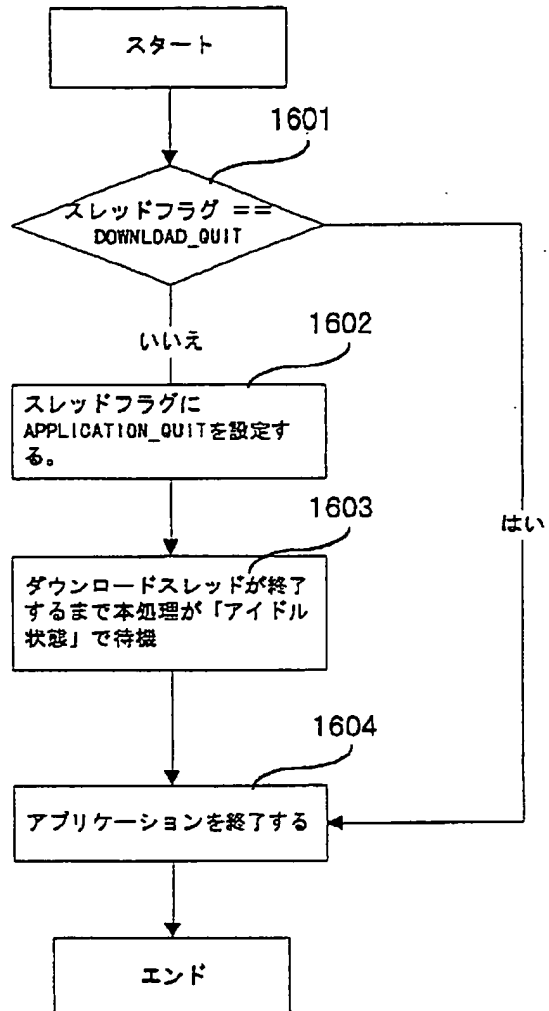


【図21】

2100

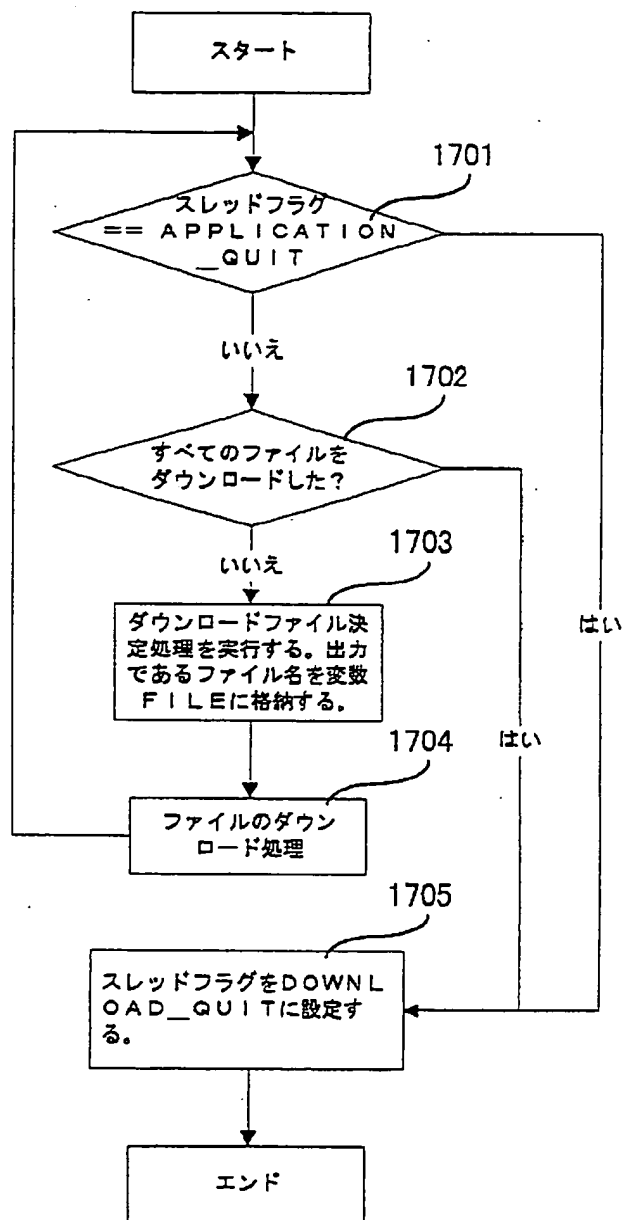
2101 モジュールファイル名称	21011 21012 依存する他のモジュールファイル名称配列
以降、同じデータ構造	

【図16】

1600

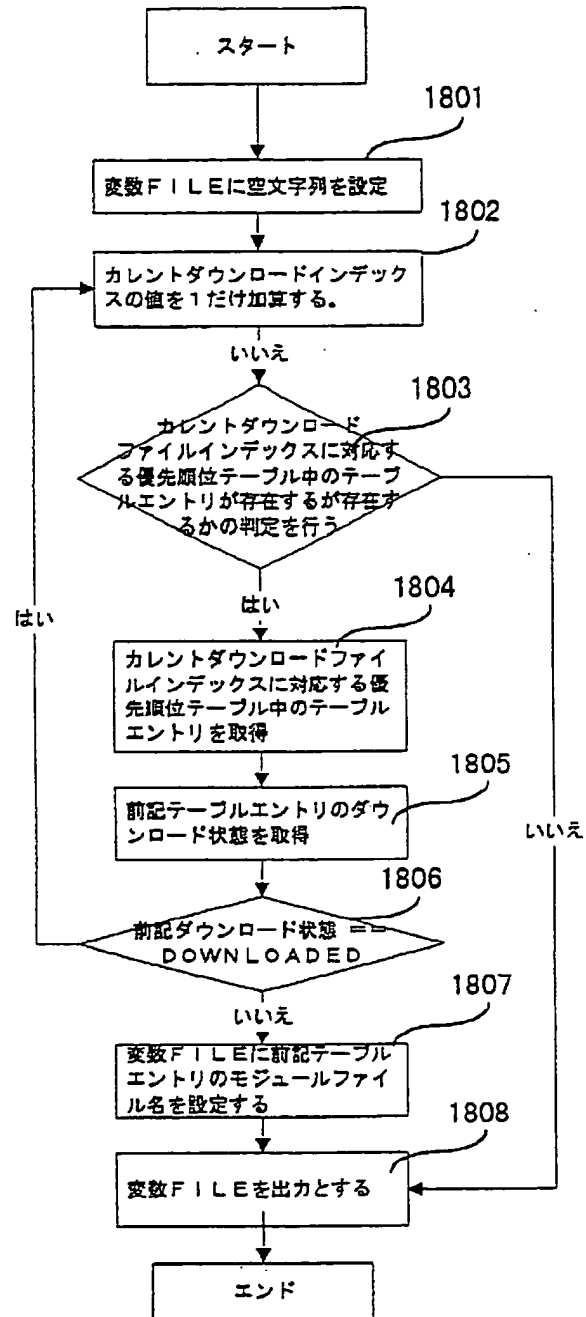
【図17】

1700

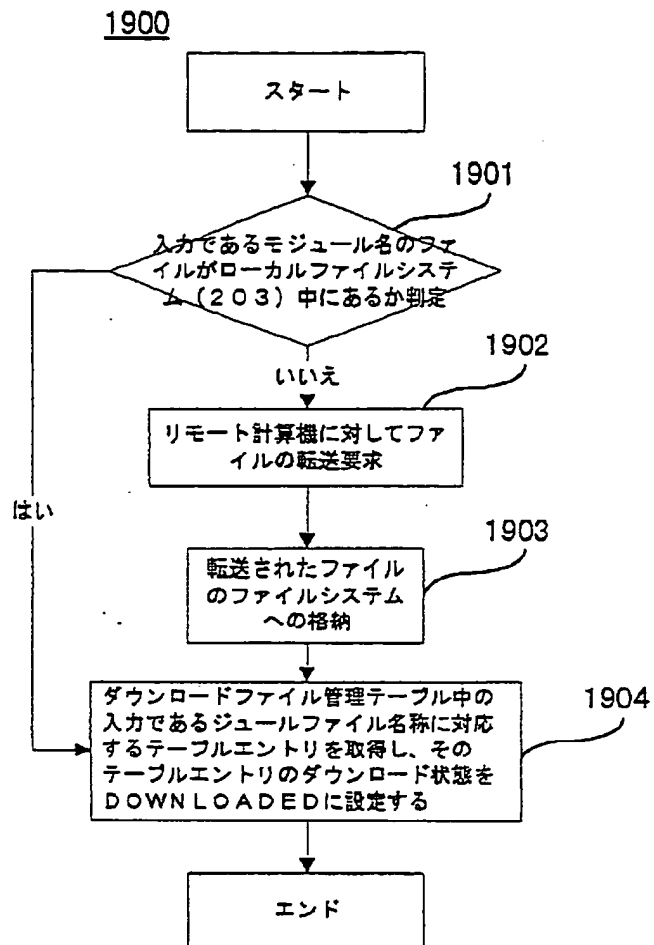


【図18】

1800



【図19】



【図22】

2200

ステップ1801

2201

文字列配列変数 $X :=$
 モジュールファイル間依存関係情報中の、リセントモジュールファイル名に対応する依存関係情報エントリのモジュールファイル名称配列
 (21012)

2202

 $n := 1$

2203

 $n \leq X$ の長さ

いいえ

ステップ1802へ

はい

2204

X の n 番目のファイル名称の
 ファイルがローカル計算機の
 ファイルシステムにあるか判定
 する

いいえ

2206

FILE :=
 X の n 番目のファイル名称

ステップ1808へ

2205

 $n := n + 1$

フロントページの続き

(72) 発明者 本田 由里

神奈川県川崎市幸区鹿島田890番地 株式
 会社日立製作所情報・通信開発本部内

(72) 発明者 嶋崎 康一

神奈川県横浜市戸塚区戸塚町5030番地 株
 式会社日立製作所ソフトウェア開発本部内